Microcomputer Controlled Hydraulic Circuits

Sengoda G. Ganesan
Northern Illinois University

ABSTRACT

Linear actuators in the hydraulic circuits can either be mechanical or Solenoid controlled direction valves. Random and computer logics are the two alternatives for electronic actuators. The actuators with computer logic can perform various sequences with programmed controls and can be very effective in terms of cost, speed, and reliability.

INTRODUCTION

Many hydraulic circuits encountered in industry require a series of linear actuators. Circuits commonly used today employ sequencing valves, mechanically operated direction valves, or solid state logic circuits. The solid state logic circuits may either be of random or of computer logic type. The random logic circuits are designed for specific (fixed) sequence requiring considerable hardware change for changing the actuator sequence. The computerized logic circuits are software controlled and therefore, the actuator sequence can be changed without any change in the hardware. With the advent of the microprocessor, many options are available for sequencing linear actuators with computerized logic circuits. Microcomputers can be used in programmed, interrupt, or Direct Memory Access I/O mode. The programmed I/O is both software initiated and software controlled, whereas the interrupt I/O is hardware initiated and software controlled.

This paper illustrates how the microcomputer can be used in a particular multi cylinder-sequencing problem in the interrupt mode. The problem chosen for illustration is that of a situation with seven different sequences. Each sequence in the cycle involves four cylinders A,B,C, and D. The arbitrarily chosen order for one of the sequences in the cycle is A+, C+, C-, B+, D+, B-, D-, A-. The first sequence in the sequence transition has the top priority. Initially, the sequence status flags for all the sequences are on. Upon power up, the microcomputer starts with the first sequence and proceeds with the priority rule established by the sequence transition within the cycle. Once all the sequences are completed, the cycle is repeated indefinitely.

THE CIRCUIT

A microcomputer controlled hydraulic circuit is shown in figure 1. On the hydraulic side, there are seven 4-cylinder sequences. The cylinders in each group are connected to 2- position 4-way, solenoid actuated, spring return directional valves. Limit switches at the actuator extremities supply electrical signals indicative of the status of each actuator. A typical sequence arrangement is shown in figure 2.

The electrical side of the circuit is shown in figures 3, 4, 5 and 6. A 8080-based microcomputer system provides the actuating signal for the solenoid. The system consists of input, output, interrupt, and processor circuits along with the appropriate software stored in memory. There are seven input ports, one for each sequence, to sense the limit switches. Each sequence consists of four cylinders and therefore, eight limit switches must be provided to monitor the cylinder position. The microcomputer system is a computer logic circuit with a common data bus for all the peripherels in the system. Since there can be no more than one active source on the bus during data transfer operations, these limit switches should be connected to the bus through a 8-bit buffer, 8212. These buffers must be controlled with device select pulses from the control bus.

The solenoids are actuated by the output ports. There are seven 4-bit latch type output ports. These latches are interfaced with the microcomputer system through the data, control, and the address buses. The output port interface for a typical sequence is as shown in figure 4. Since there are more than one input and output ports, the device select pulses for the I/O ports must be generated by a decoder using the control and the address buses as shown in figure 5.

The microcomputer system is built around an 8080 microprocessor. The cpu has a 16-bit address bus and an 8-bit data bus. This permits the processor to handle up to 65k bytes of memory, 256 inputs ports, and 256 output ports. The most important characteristics of the processor is its interrupt capability. The cpu can be used in programmed, interrupt, or in the Direct Memory Access mode. The mode selection can be selected on the 'Fly' and to make the decision for selecting the mode, one has to be aware of the fact that the programmed I/O puts the cpu in a timing loop thereby reducing the data transfer rate. In the case of interrupt I/O, the cpu is switched between different programs by a hardware signal. This enables the system to have a higher data transfer rate. This feature in the interrupt I/O mode demands an extensive switching circuits. Therefore, the mode-decision is to make a trade-off between hardware switching circuit cost and the data transfer rate.

The interrupt interface begins with seven sequence status flags representing the status of the sequences. These flags are implemented with seven flip flops. The logic^{-1} content of a flip flop indicates that the appropriate sequence needs a complete sequencing or the current sequence is incomplete. Whereas, a logic-0 indicates that the appropriate sequence is complete and the cpu is currently processing the down-stream sequences in the cycle. At the beginning of the cycle, the processor sets all the flags at logic 1. Every time a sequence is completed, the appropriate flag is reset.

The status flags are interrupt request devices requesting the processor for sequencing service. The sequence transition diagram for the example problem is as shown in Fig. 8. The sequences in the cycle are of different types. For example, they may represent operations such as drilling, and milling with different machine tools. Therefore, the cpu is confronted with the problem of device identification. This can be alleviated either by a polled approach or by a hardware circuit. In this example, a priority encoder circuit is used to implement the sequence transition as in Figure 8. The encoder generated a 3 bit output and an interrupt request signal according to the inputs from the status flags. The 3-bit binary number represents a RST instruction and is interfaced with the system through an 8212 tri-state buffer using the data and control buses. The interrupt request output of the encoder is connected to the INT line of the processor.

CIRCUIT OPERATION

The software for the circuit operation must include eight service routines or traps. The first routine is called the power-up routine. When the system is powered up, the processor starts with the power up routine. The power up routine, figure 7, defines the stack memory, sets the sequence status flags, enables the interrupt, and waits for interrupt at the halt mode. Since all the status flags are set, the encoder generates the interrupt request signal and RST-1 output. The cpu generates INTA control signal to turn on the 8212 buffer in the interrupt circuit. When RST-1 instruction reaches the cpu, it switches the cpu into the trap #1. In the trap, the processor performs the sequencing, resets the appropriate status flag, returns back to the power up routine, and waits for interruption. Since the upstream status flag is disabled, the next output from the encoder will be RST-2. The cpu, now, will be switched into the trap #2. This logic will be repeated until the cycle is completed. During the last sequence in the cycle, the cpu will set all the flags before returning to the power up routine.

The flow chart, shown in figure 9, describes a typical trap logic. At the beginning of the trap, a memory pointer is set to address the location in memory where the first command word is stored. The processor reads the first command word in memory and loads the last four bits of it into the

output port. In the example sequence, the command word would be XXXX1000. This command word produces a positive drive voltage on solenoid A, and zero voltages on solenoids B,C, and D. This actuator A is commanded to extend, while B,C, and D remain retracted. The cpu then reads the input port to ascertain the status of the actuators. By comparing the feedback signals to the command word it can determine whether or not the most recent command word has not been carried out, so the processor waits until the difference goes to zero by looping backward. When the feedback word equals the command word, the cpu tests whether the end of the sequence has been reached by checking whether the memory pointer has been incremented eight times (the example sequence requires eight command words). If the pointer has been incremented eight times, the processor resets the flag and returns to the power up routine.

In the last sequence in the cycle, upon incrementing the pointer eight times, the cpu also sets all the flags for repeating the cycle. The list of command words for the example sequence is given in Table 1.

CONCLUSION

In the example problem an 8 bit processor was chosen. It can handle only 256 input and 256 output ports. If a complex cycle is to be controlled, one may have to explore the possibility of using processors that can handle more ports. The advantage of the computerized actuators is that they can perform various sequencing functions without hardware change. This feature may enable better utilization of hardware on a functional basis as compared with random circuits. The designer has to be alerted to the fact that the computer logic is not always the solution. The choice between random and computer logic should be based upon the number of sequences, speed, reliability, and cost.

REFERENCES

1. *Hydraulic Control Systems,* H.E. Merritt, John Wiley & Sons, 1967.
2. *MCS-80 User's Manual,* Intel Corporation, 1977.

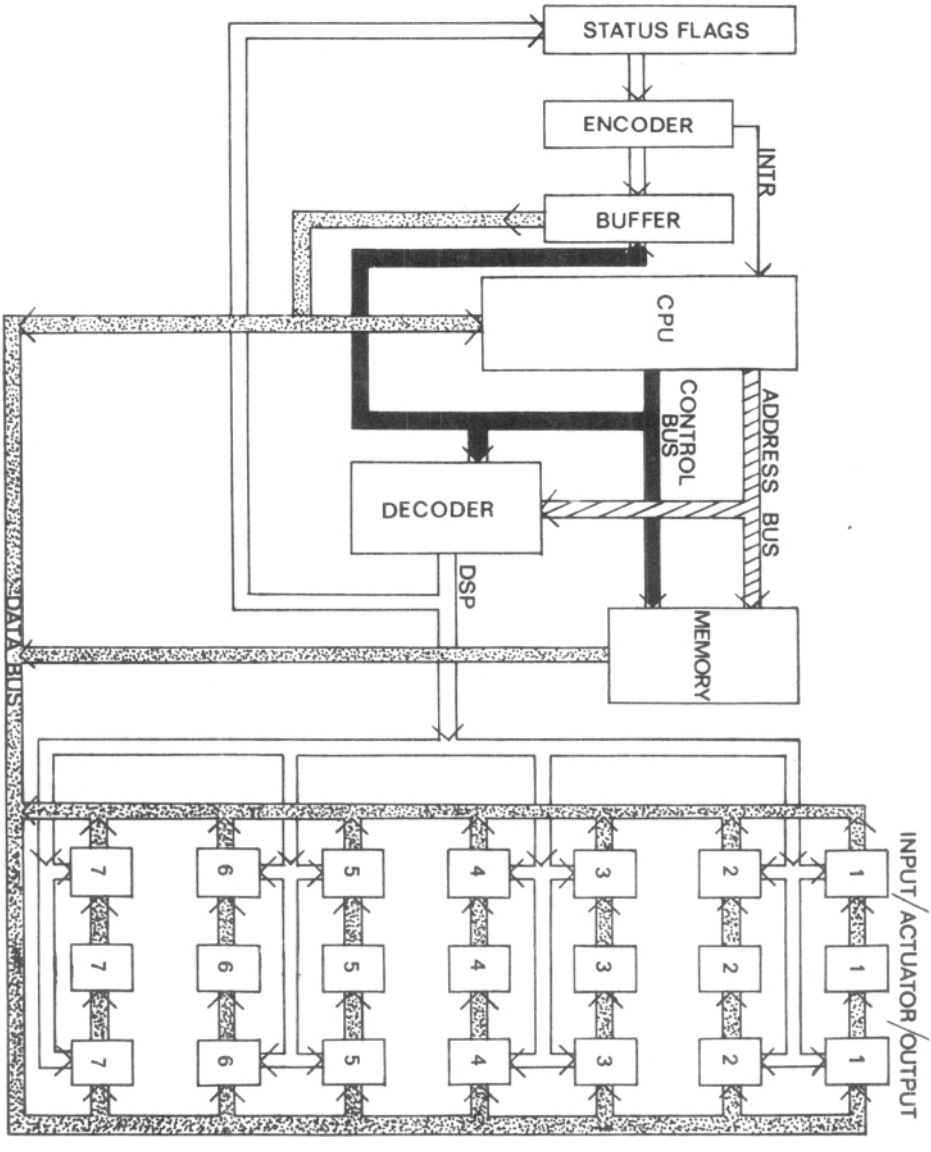

FIG. 1 MICRO CONTROLLED HYDRAULIC CIRCUIT

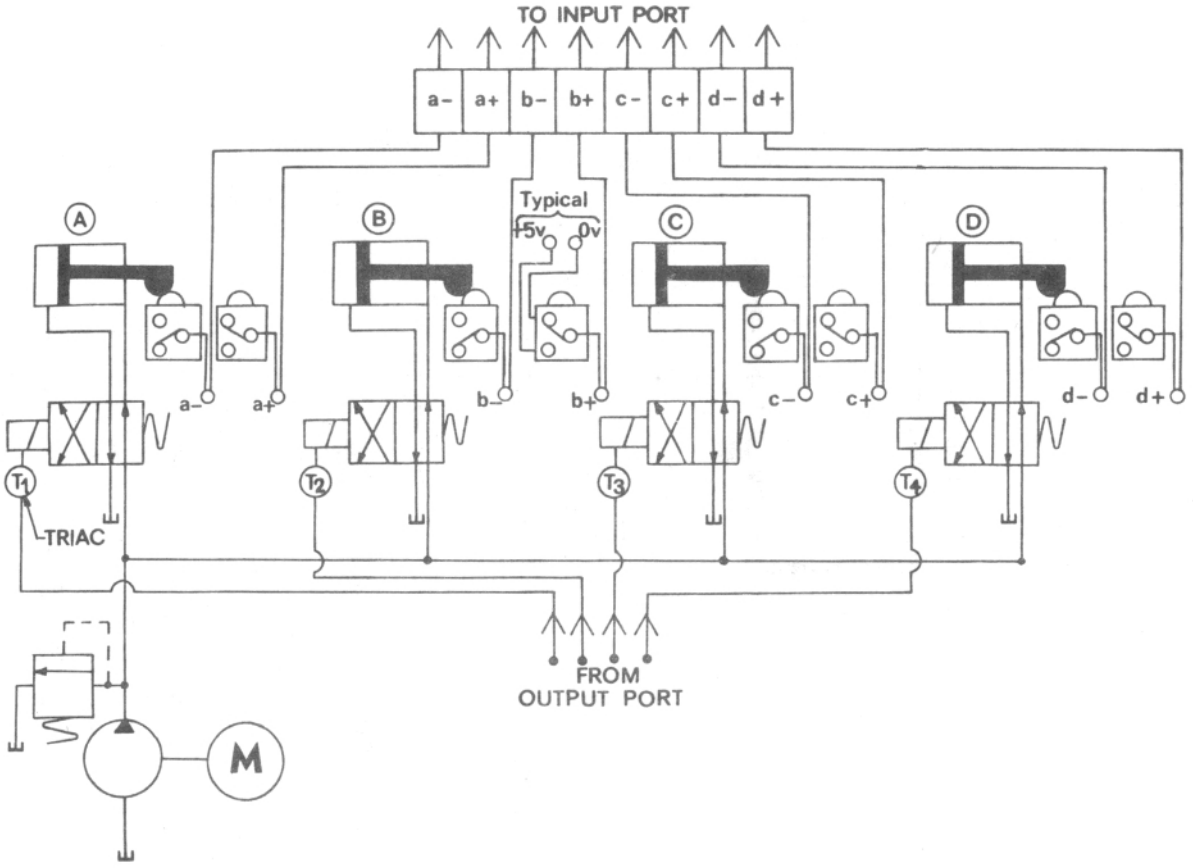

FIG. 2 HYDRAULIC CIRCUIT

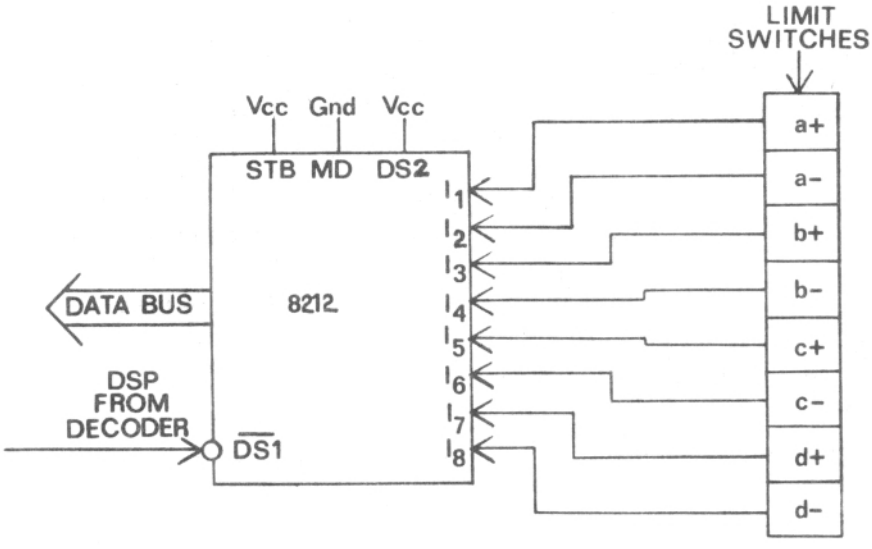

FIG. 3 INPUT INTERFACE FOR A TYPICAL SEQUENCE

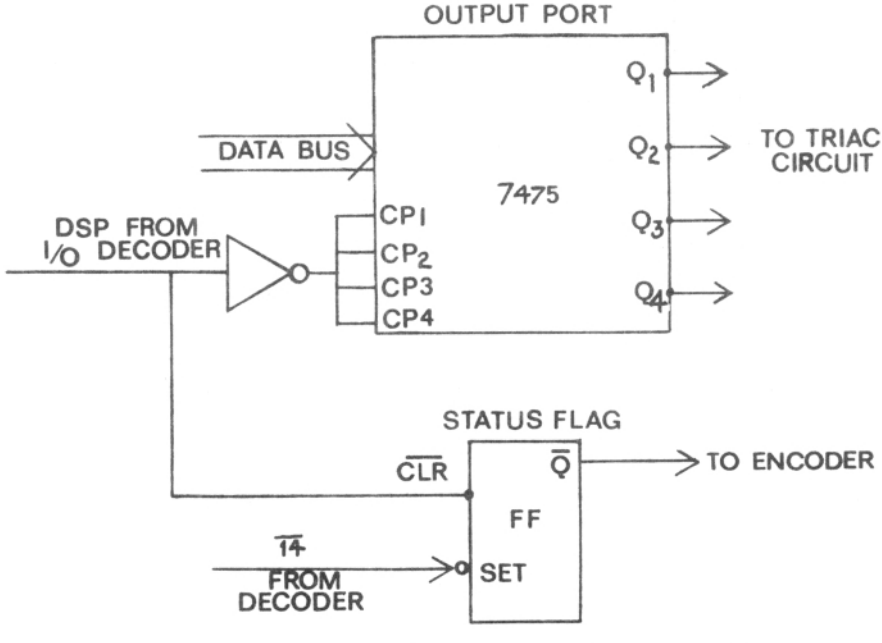

FIG. 4 OUTPUT INTERFACE FOR A TYPICAL SEQUENCE

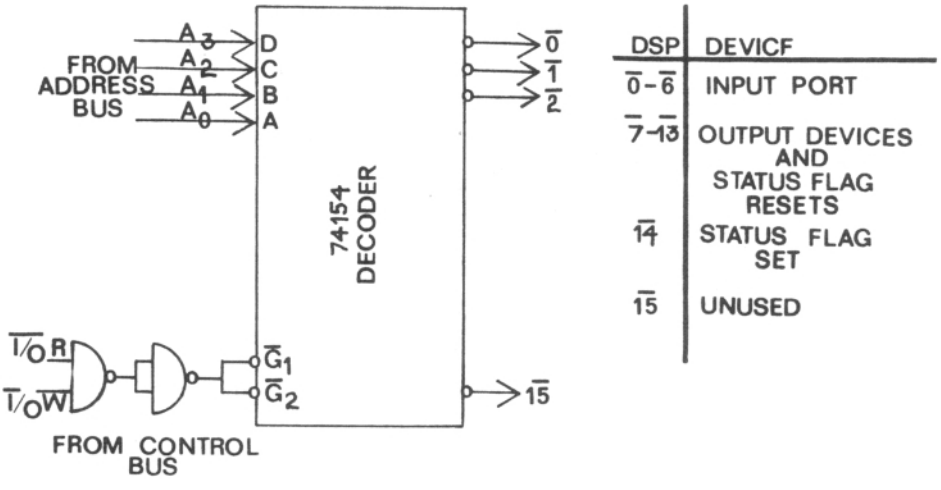

FIG. 5 I/O DECODER

COMMAND NO.	a+	a-	b+	b-	c+	c-	d+	d-
1	1	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	1	0	0	0	0	0
5	1	0	1	0	0	0	1	0
6	1	0	0	0	0	0	1	0
7	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

TABLE .1 COMMAND WORDS FOR EXAMPLE SEQUENCE

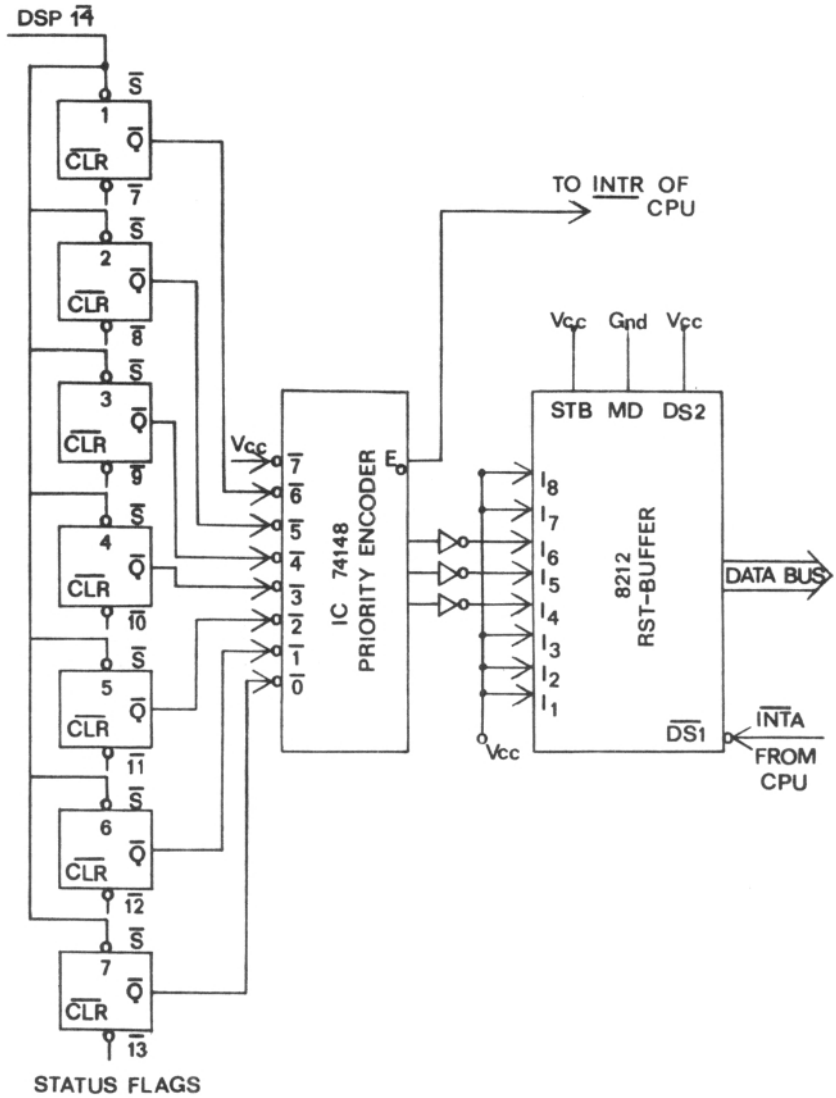

FIG.6 INTERRUPT INTERFACE

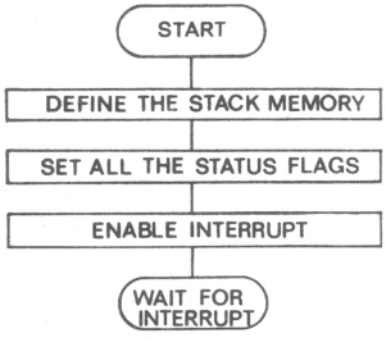

FIG. 7 FLOW CHART FOR POWER-UP ROUTINE

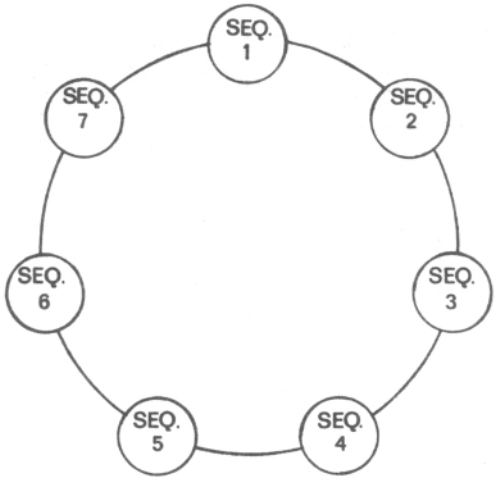

FIG. 8 SEQUENCE TRANSITION IN THE CYCLE

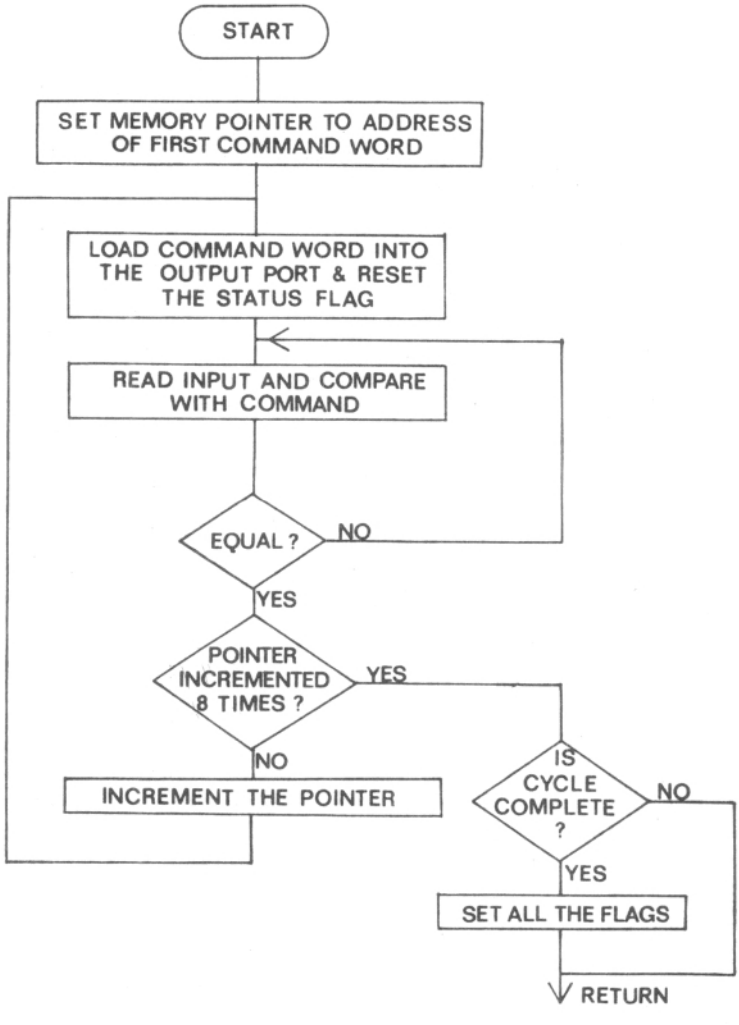

FIG. 9 FLOW CHART FOR TRAP ROUTINE